Large scale analysis of HTTP adaptive streaming in mobile networks

Ali Gouta[1,2], Charles Hong[3], Dohy Hong[3], Anne-Marie Kermarrec[2] and Yannick Lelouedec[1]

[1]Orange Labs,{ali.gouta, yannick.lelouedec}@orange.com
[2]INRIA Rennes Bretagne-Atlantique,{ali.gouta, Anne-Marie.Kermarrec}@inria.fr
[3]N2Nsoft,{charles.hong, dohy.hong}@n2nsoft.com

***Abstract*— HTTP Adaptive bitrate video Streaming (HAS) is now widely adopted by Content Delivery Network Providers (CDNPs) and Telecom Operators (Telcos) to improve user Quality of Experience (QoE). In HAS, several versions of videos are made available in the network so that the quality of the video can be chosen to better fit the bandwidth capacity of users. These delivery requirements raise new challenges with respect to content caching strategies, since several versions of the content may compete to be cached. In this paper we present analysis of a real HAS dataset collected in France and provided by a mobile telecom operator involving more than 485,000 users requesting adaptive video contents through more than 8 million video sessions over a 6 week measurement period. Firstly, we propose a fine-grained definition of content popularity by exploiting the segmented nature of video streams. We also provide analysis about the behavior of clients when requesting such HAS streams. We propose novel caching policies tailored for chunk-based streaming. Then we study the relationship between the requested video bitrates and radio constraints. Finally, we study the users' patterns when selecting different bitrates of the same video content. Our findings provide useful insights that can be leveraged by the main actors of video content distribution to improve their content caching strategy for adaptive streaming contents as well as to model users' behavior in this context.**

General Terms:
Measurement, Design

Keywords:
HTTP Adaptive Streaming, HTTP Live Streaming, Microsoft Smooth Streaming, Live, Catch-up TV, chunks, profile, access networks

I. Introduction

The proliferation of mobile smart phones is significantly changing the landscape of mobile usage. New smart phones are designed for the best high definition video experience and are now commonly used by users to watch TV programs either in live or in replay. In [1], Cisco reported that mobile video will grow at a compound annual growth rate of 90% between 2011 and 2016. All these factors suggest that mobile streaming services will soon dominate the mobile communication landscape. Hence, ensuring a fast and a reliable content delivery accounting for users' network constraints is key to this evolution.

A new HTTP-based strategy, called HTTP Adaptive bitrate video Streaming (HAS) has recently emerged, which takes up this challenge. Adaptive streaming over HTTP encodes the video stream with several encoding bitrates, thus generating several presentations of the same video content. Each presentation is then segmented into smaller parts, called chunks or segments, and usually varies between 2 to 10 seconds length. When a user requests a video content, the hosting server sends back to the client a formalized description of the media presentation, named manifest file or Media Presentation Description(MPD), depicting all available bitrates of the requested content. The MPD enables the client-player to choose the quality that matches best her requirements, network and device capabilities.

HAS is widely used in mobile TV industry. Usually, TV broadcasters delegate to CDNs and to streaming servers the video segmentation process and to generate the adequate manifest file. CDN providers and Telcos are the most concerned to ensure a reliable and fast end to end delivery between users and servers. Therefore, studying HAS properties from an operator standpoint enables to accurately understand the implications of such a way of delivery on users' behavior. As important as it is, this scope is not yet as well studied by the research community as others are (*e.g.* User Generated Content [2] or IPTV systems [3]). In this paper, we conduct an in-depth analysis of the behavior of mobile clients when requesting such HAS streams using a large scale dataset. We also study through simulations, the opportunity to enhance the caching of such chunk-based streams within proxy caches.

We believe that the outcome of our analysis can be leveraged to help content providers design caching strategies to improve their services. The rest of the paper is organized as follows. Section II presents our dataset. In Section III, we focus on the chunk-based delivery properties, we propose a fine-grained definition of content popularity and we study through simulations the caching implications when requesting such chunk-based contents. In Section IV we study the impact of the network environment on users' behavior

Part of the research leading to these results has received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n: 248775.

when requesting HAS contents. In Section V, we analyze the distribution of the selected video bitrates by the client-players. Finally, we conclude in Section VI.

II. Data collection and processing

Our dataset has been collected from five Measurement Points (MPs) spread within France. More precisely, the measurement points are located in the mobile backbone on 5 Gi interfaces just above the Gateway GPRS Support Nodes (GGSNs) as shown in Figure 1. That way, we collect all HAS sessions of all mobile subscribers for the considered operator in France. The measurement system is based on a passive observation of the IP traffic of mobile customers. The data collection is based on capturing all packet headers of HAS streams that contain useful information, such as the packet size and sequence number. Confidentiality and privacy are preserved as we do not have access to the payload data and all clients' and subscribers' identifiers are anonymized. We aggregate all information relative to each persistent-TCP connection and export it to a database, from where it is analyzed. Each persistent-TCP connection corresponds to one downloaded video segment. This means that the number of downloaded chunks during one HAS session is equal to the number of persistent TCP connections established between the server and the client over a period of time.

The measurements were conducted over a 6 week and one day period, from February 28^{th} to April 10^{th} 2012, involving $485,544$ unique active clients and $8,131,747$ HAS sessions.

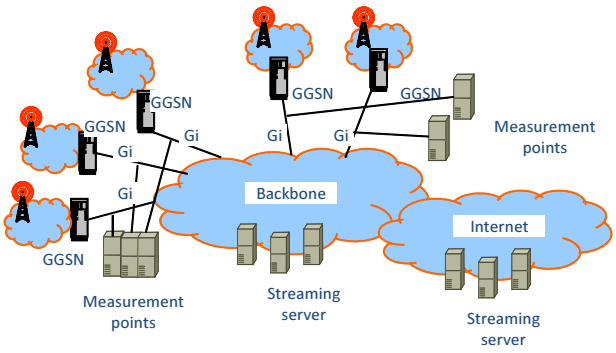


Figure 1. Streaming infrastructure and data collection

Our dataset mainly contains:

- Live sessions: where clients watch live TV;
- Catch-up TV sessions: where TV content providers allow clients to replay a set of videos previously broadcasted in live.

In Table I, we show the breakdown of the live and catch-up TV HAS sessions collected over the 5 MPs. On average, we observe that around 70% of the HAS traffic corresponds to live TV streaming sessions. The dataset encompasses 8,127,762 Apple HTTP Live Streaming (HLS) sessions and 3,985 Microsoft Smooth Streaming (HSS) sessions. Therefore, in the remainder of this paper, we focus on the Apple HLS traffic as it represents the very large majority of the overall collected HAS traffic.

	Live sessions	catch-up TV sessions
MP 1	606,470 (63%)	354,025 (37%)
MP 2	1,198,577 (68%)	538,936 (32%)
MP 3	1,333,619 (71%)	528,745 (29%)
MP 4	1,281,758 (72%)	483,185 (28%)
MP 5	1,290,546 (71%)	515,886 (29%)

Table I
Percentage of Live and catch-up TV sessions

We identify a session by the first HTTP-GET request of the first chunk for which the URL address ends either with a *.ts* for a HLS stream, or with a *.ism* for a HSS stream. Each HAS session corresponds to the set of chunks sent to the associated client over a period of time. Then we classify the requested chunks according to their encoding bitrates (profiles). The encoding schema of the different profiles of video contents may differ slightly from one TV content provider to the other. Hence, we proceed by setting a scale of 8 ranges (See Table II) that closely fit the users' profiles mostly recommended by Apple [4].

To map a given requested chunk to the associated profile, we assume that its size (in bytes) is approximately equal to the volume of information contained in the HTTP payloads transporting it, and that its encoding bitrate *(video bitrate + audio bitrate)* is equal to its size divided by its duration. The MP_i starts measuring the size of the chunk once it detects the HTTP-200 response for that chunk's request. We get the duration of the chunk from the manifest file relative to each video session.

Profile i	Encoding bitrate (kbps)
Profile 0	< 50
Profile 1	[50-150[
Profile 2	[150-280[
Profile 3	[280-420[
Profile 4	[420-600[
Profile 5	[600-1000[
Profile 6	[1000-2000[
Profile 7	≥ 2000

Table II
Profiles

Then, we classify all information relative to the requested chunks belonging to the same session as per their profile. The following data model shows in a tree hierarchy the most relevant fields from our data set for the present study:

Session
Start (in timestamp).

End (in timestamp).
content_id.
client_id.
$for\ i \epsilon [0..7]$.
profile$_i$:
Number of chunks requested in profile$_i$.
Bytes downloaded in profile$_i$ including packet header.
Bytes downloaded in profile$_i$ without packet header.
time to deliver all requested chunks in profile$_i$.

III. Chunk based analysis

Analyzing video popularity from different streaming technologies has always been a major focus by the research community. It was always reported that few of videos get the most interest of the clients while the majority are less viewed. This suggests for instance that caching only the most popular videos is enough. This phenomenon has been already formally described by the Pareto principle. While, our data set confirms once again this property for the catch-up TV videos, we believe that the segmented nature of catch-up TV requires a finer grained analysis of content popularity i.e. at the granularity of one chunk. We therefore, introduce and analyze the following two dimensions: *video popularity* and *chunk popularity*. The former one refers to the number of times, end-users requested a given video in a given period of time, and the latter one refers to the number of times clients requested a given chunk in a given period of time.

A. Popularity of catch-up videos across the time

We analyze in Figure 2 the access frequency measured at a granularity of one hour of the top-0.6%, top-1%, top-5%, top-17% and top-100% most requested catch-up TV videos over the whole period of data collection. For instance, in order to compute the access frequency of the top-5% most requested catch-up TV videos at day 7, we concentrate on the 5% catch-up TV videos which were most requested over the whole period of data collection; we then sum the access frequency of the aggregate number of requests for each of these videos at the 7^{th} day after their respective release date. We observe that the popularity of catch-up videos is age-sensitive and that clients' requests fade significantly as time goes on. The number of requests for such contents decreases with a ratio of 1/8, just 2 days after putting them available to the mobile audience. The popularity profile of these catch-up TV videos on mobile devices is significantly different from other types of video applications, in particular User Generated Content (UGC) applications (e.g. Youtube) and Premium VoD services (e.g. NetFlix) where user preferences are globally insensitive to the most popular videos [2].

Generally catch-up TV videos are contents that were broadcasted in live TV for the last week or at best for the last month, due to rights negotiation with TV content provider. Although the access frequency decreases over time, the limited availability period is still constraining for the end users.

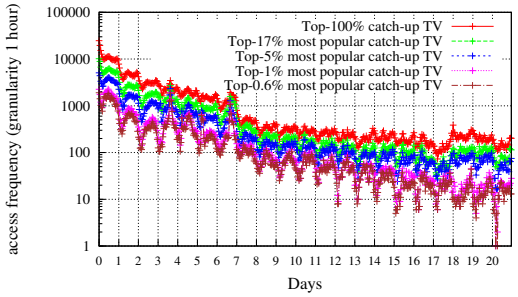


Figure 2. Popularity over time of the most requested videos

We observe in Figure 2 that the access frequency decreases with a ratio of 1/10 after seven days left. However, it further decreases with a ratio of 1/40 the next day. This is explained by the fact that some of the content providers purge the 7-days old videos from their catalogue. Another reason that makes catch-up videos loose popularity so rapidly is that viewers are not likely to watch the same video multiple times as they do for mutable web objects. This *Fetch-at-most-once*-like behavior was initially introduced by Gummadi et al. in [5] while trying to understand the file popularity paradigm in P2P downloads. Throughout our dataset, we validate this suggestion for catch-up TV contents and we find that in average 82-percentile of clients do request at most one time the catch-up TV video, while for around 97-percentile of clients do request the same content at most 3 times.

B. Chunk popularity and chunk based analysis

Popularity of a given chunk is dependent on two parameters: the popularity of the video it belongs to and the position of this chunk within the video stream: Clients may abort their sessions or make a jump forward/backward when watching the video. More formally:

Let m_i be the number of chunks within video i.

Let p_j^i be the probability of requesting $chunk_j$ ($j \epsilon [1 : m_i]$) within a video i.

Let $q_{l,j}^i$ be the conditional probability that the next request of the client who just requested $chunk_l$ will be for $chunk_j$, either by jumping or moving forward.

Let $s_{j>0}^i$ be the probability of aborting the session after requesting j chunks.

Then we have the following equation:

$$\begin{aligned} p_j^i &= \sum_{l=1}^{m_i} p_l^i * q_{l,j}^i \\ &= p_{j-1}^i * q_{j-1,j}^i + \sum_{l=1, l\neq j-1}^{m_i} p_l^i * q_{l,j}^i \end{aligned} \quad (1)$$

From now on, we assume that clients never make a jump forward/backward when watching the video[1]. Equation 1 becomes:

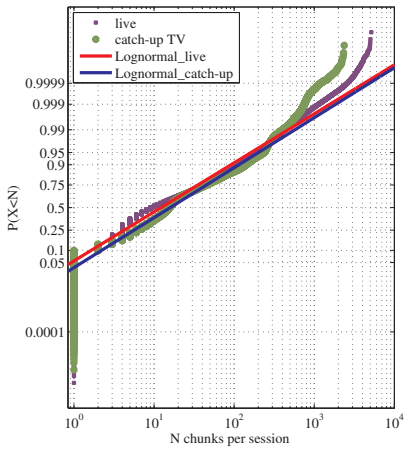
[1]Previous studies on video streaming in mobile context show that up to 80% video sessions are without any trick mode (no pause and no jump forward/backward) [6], [7]

$$p_j^i = p_{j-1}^i * q_{j-1,j}^i = \begin{cases} p_1^i;\ if\ j = 1 \\ p_1^i * \prod_{k>1}^{j} q_{k-1,k}^i = p_1^i * (1 - s_j^i);\ if\ j > 1 \end{cases} \quad (2)$$

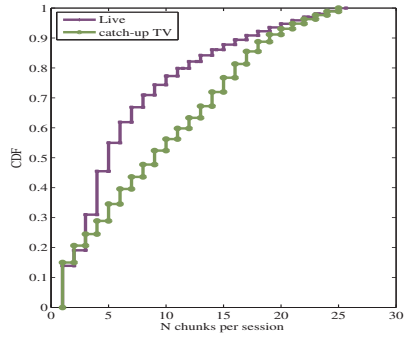
p_1^i represents the probability that clients request the first chunk of a given video i within the catalogue. In other words, it represents the probability of selecting video i, when the client browses the catalogue.

$(1 - s_j^i)$ is the probability of requesting at least the j first chunks once video i is selected.

Using our dataset we can estimate s_j^i statistically by determining the law that best fits the distribution of the number of requested chunks per HAS session. Figure 3(a) represents the cumulative distribution function (cdf) of the number of requested chunks per HAS session, for both live TV and catch-up TV videos.



(a) CDF of number of chunks per HAS session for Live and Catch-up TV sessions



(b) CDF of HAS sessions where clients request less than 25 chunks

Figure 3. CDF of the requested number of chunks per HAS session

Using the maximum likelihood (MLE) estimation, we determined that the log-normal distribution laws are best matching the respective cumulative distribution functions (cdf) of the live TV and catch-up TV video sessions. Then we used the Kolmogorov Serminov goodness-of-fit test to assess the accuracy of the estimated parameters characterizing these log-normal laws. We set the confidence level to up to 95%. The KS test is well adapted here given the high number of samples of the catch-up TV sessions and Live sessions within the dataset. The estimated parameters of the log-normal laws and the results of the KS test are provided in Table III.

We can make several observations about Figure 3(a). First, the first chunks in the temporal sequence of a given content are much more requested than the last ones. We observe that around 90% of video sessions does not exceed 100 chunks, *i.e.* 16 min (each chunk containing a 10 second of the video).

Many factors influence the sessions' duration. Yet the most important ones are obviously the billing model and the nature of the client device. In our case, as the video application is free or included in a flat rate mobile package, clients do not hesitate to stop and switch between contents before the end. Moreover, clients' mobility (*e.g.* being in public transport) may also be leveraged by shortening sessions' length. Clients seem to privilege alternative options for longer video sessions (TV receiver or PC connected via satellite, terrestrial, cable or fixed networks). For instance, we find that only 20% of the video sessions exceed 10 min in our dataset, whereas authors reported in [8] that more than 47% sessions exceed that same duration in the Power Info DVD Systems which contain catch-up TV videos but also a set of premium feature films.

Second, the cumulative distribution functions diverges from the log-normal laws when the HAS sessions have more than 1000 chunks. A minority of clients keeps on requesting chunks over a much longer period than the others, and some until the end of the video. In the case where the chunks are delivered by edge servers (belonging for example to a CDN or a set of transparent caching servers) such a tailed profile could degrade the caching efficiency of these edge servers, even more if their cache size is limited and their caching replacement algorithm is highly reactive to the end user requests. For instance, if an edge server is configured with the LRU caching algorithm (Least Recently Used), any request for a chunk located at the end of a video will make this server caches this chunk and sets it at the top of its LRU ranking. Then, even if there is no further request to that chunk, it will remain for a while in the cache before being purged, possibly at the expense of some other, more frequently requested, chunks. One could therefore investigate the interest of adapting caching algorithms so that they do not cache the last and very infrequently requested chunks in videos: One could extend the object replacement algorithm used by the edge servers by setting a *chunk-position threshold*, beyond which the edge servers would not cache any chunk. The section III-C presents the results of a trace-driven simulation enforcing the relevancy of such

a policy.

Third, the range where the cumulated distribution functions of the live and catch-up TV HAS sessions differ most is on the first 25 chunks of the sessions. We zoom on this range in Figure 3(b). Clearly, the density of Live sessions within the range 1 to 5 chunks per Live session is much higher than the density between 6 to 25 chunks per Live session. This could be explained by the fact that users spend more time in *discovering* or *recognizing* the program broadcast in live. Then, they loose rapidly the interest in watching the live channels; they rather give up or switch to another channel. However, for catch-up TV videos, we find that for the considered range, clients spend more time watching video contents than watching live channels. This explains why the CDF of the catch-up sessions is below the CDF of Live sessions. At this stage, it is important to note that if a client aborts her session after requesting the fifth chunk, this does not mean that she indeed spent 50 seconds (assuming that one chunk is 10 seconds length) watching the video as the chunks are requested and buffered in advance before being displayed. Previous works focused on assessing the specific properties of well-known implementations of HAS clients, including Netflix, Adobe dynamic streaming, Apple HTTP live streaming client (HLS), and Microsoft smooth streaming clients, in particular regarding start-up time and playback buffer [9]. At the start, the playback latency requires buffering to up to 30 seconds of the video stream in the case of HLS. This means that the client-player buffers up to 3 chunks before starting playing in order to guarantee a smooth playback quality. Figure 3(b) shows that it could be worth investigating the opportunity to adjust the client-player buffering rules and the edge servers' caching replacement methods to the type of content delivered (Live TV content vs. Catch-up TV content).

	σ	μ	test statistic	p_value
Lognormal_live	1.57033	2.50498	0.1019	0.2368
Lognormal_catch-up TV	1.54	2.76728	0.0839	0.4648

Table III
ESTIMATED PARAMETERS

C. Caching implications

Content popularity distribution may have a strong impact on the performance of content distribution systems. In distributed systems such as CDNs (Content Delivery Networks) this is a key factor influencing the *cache hit ratio* - *i.e.* the percentage of requests successfully handled by the CDN edge servers -, and thus the network resource utilization, the delivery cost and the quality of service experienced by the end user. With a trace-driven simulation we now illustrate the implications of caching chunk-based streams and investigate the benefit of setting a *chunk-position threshold* as defined in Section III-B.

The trace-driven simulation uses a seven-day trace from the collected HAS dataset and encompasses a $29,921,935$ HTTP-client requests. The trace file contains information about the timestamp of each new session, the video ID (we associate an id to each video content within the data set), and the number of chunks requested by the clients in each session. Then we simulate the scenario where all HAS traffic gets forwarded to a proxy-cache deployed just after the 5 GGSNs. For the sake of simplicity the simulation relies on the four following assumptions:

- The videos are encoded at 500kbps, *i.e.* with a single profile, profile 4, as it is the most common requested profile between clients and where the sojourn time is largely superior than profile 2 and 3 as we will detail in Section V.
- All chunks are 10 second length.
- We only consider the Catch-up TV sessions. Effectively in Live sessions, chunks are generated on the fly. Hence, this introduces some complexity on assessing the time between generating the chunk on the fly, caching it and the difference in time between clients watching the same content in live. This will be the subject of future work.
- Clients do not make any jump forward/backward during the video session.

1) LRU with Chunk position threshold: Figure 4 represents the cache hit ratio obtained with this simulation scenario for different cache sizes (C) and for different values of the chunk-position threshold (C_{th}).

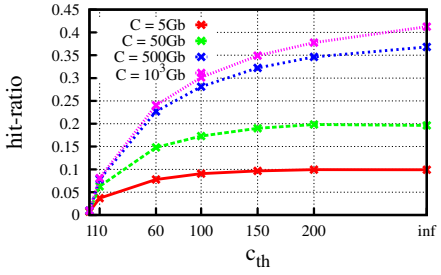
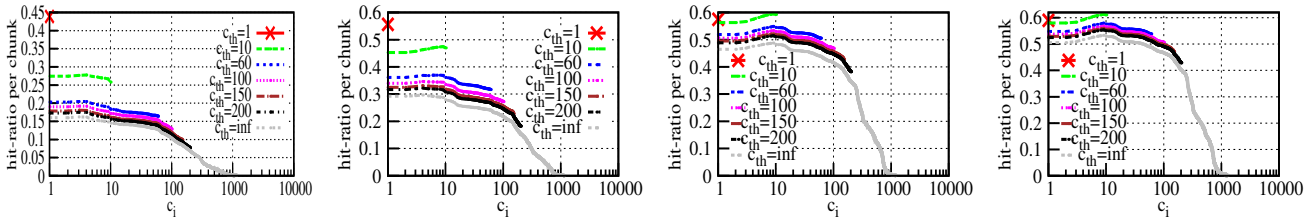


Figure 4. Cache Hit-ratio with respect to c_{th}

Figure 4 shows that for small cache sizes (C<50Gb) setting C_{th} to 100 leads to approximately the same cache hit ratio as if there were no chunk-position threshold (*i.e.* $C_{th} = \infty$). On Figure 5 we observe that this allows to save up to 50% of cache-update ratio (*i.e.* the ratio of requests making the cache updating the list of cached objects due to a cache-miss). For larger cache sizes (C>50Gb), setting C_{th} to 200 leads to approximatively the same cache hit ratio as if there were no chunk-position threshold *(i.e.* $C_{th} = \infty$*)*, while this allows to gain 20% on the cache update-ratio. This can contribute to significantly reduce the cache replacement processing time, especially in large caches where the object



(a) C = 5 Gb (b) C = 50 Gb (c) C = 500 Gb (d) C = 1000 Gb

Figure 6. Hit-ratio per chunk position

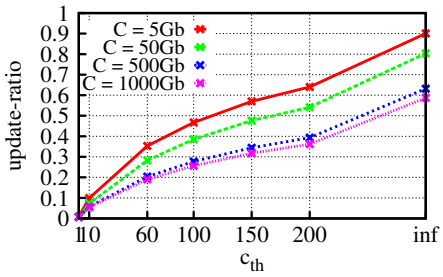


Figure 5. Cache Update-ratio with respect to c_{th}

lookup time is critical.

Figure 6 depicts the average cache hit-ratio for each position of the chunks in the temporal sequence of the videos they belongs to (*i.e.* the average cache hit-ratio for each $chunk_i$ during the whole simulation). We call this the cache hit ratio per chunk position, and we note c_i the position of the i^{st} chunk from the content in the dataset. Setting a chunk threshold (c_{th}) improves the per chunk hit-ratio for all $c_i < c_{th}$ while keeping approximately the same overall hit-ratio when setting the most appropriate threshold with respect to the cache size as shown in Figure 4. The gain per chunk hit-ratio, may reach up to 20% if we consider caching the very first chunks (*i.e.* $c_{th} = 10$). When setting the threshold to up to 60 chunks per session (80% of catch-up TV sessions as shown in Figure 3(a)) we may improve up to 5% the average hit-ratio for all $c_i < c_{60}$. This significantly increases the number of times a client is served from the cache than from the origin server.

2) LRU versus CC: In [10] authors proposed a caching algorithm, named *Chunk based Caching (CC)*, with the same objective of improving the caching efficiency and the QoS experienced by the clients by taking into account the segmented nature of HAS contents in the caching logic. The CC algorithm takes into account the time structure of the chunks belonging to the same video object by considering that if a client is requesting chunk C_i within a HAS session, she will be very likely to request chunk $C_{n>i}$ shortly. Therefore the idea was to give priority in the cache to the chunks that should be requested shortly given the latest clients' requests. This is well adapted to pay-per-view video services as the end user is motivated to watch the video she paid for, until the end. This was the simulation setup chosen by authors in [10]. In contrast, the dataset considered in our present study corresponds to free or all-inclusive video services and it shows a very different structure, with most HAS sessions ending long before the end of the videos. In Figure 7, we keep only on the 10% most popular videos of the previous simulation setup, we also keep on the parameters chosen by authors in [10].

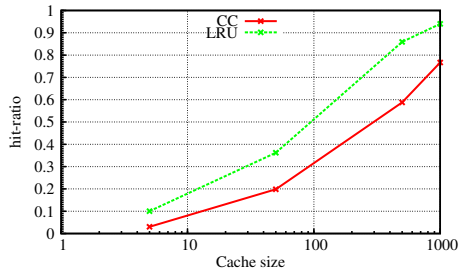


Figure 7. CC VS LRU

CC is a tunable algorithm: selecting the appropriate parameters depends basically on the nature of the traffic to be cached. We show in Figure 7 that LRU outperforms CC for the considered parameters. This is due to the fact that CC supposes that the probability to abort a session is null. Hence, it highly scores the advanced chunks position where in the reality most of the clients abort their sessions, shortly after having started viewing them. Therefore, advanced chunks position would persist in the cache regarding their high score which limits the opportunity to the first chunks of video contents to be cached. This calls for deeper investigation on how to choose the most suited parameters for CC in order to ensure a better hit-ratio.

IV. Radio Implications in HAS Streaming

In HAS, client-players try to adapt at best the playback quality to match the available bandwidth. Authors in [9] studied throughout experiments the behavior of the buffer under bandwidth restrictions using open source tools. However, in real world and in the context of mobile commu-

nication, radio conditions are potentially the main reason that pushes client to decide on the bitrate to request and to impact the time needed to deliver chunks to the end users. We assess this time in our data set, and we find that in 76% of HAS sessions, the average download time of one chunk takes less than 10 seconds. However, when chunks are 10 second long, spending an average delay of more than 10 seconds per session to deliver one chunk to the end user, would certainly turn the client-player into buffering mode. In turn, this could impact the quality of experience and triggers the HAS adaptation to lower profiles. In this section we investigate the radio implications when requesting HAS sessions. We limit our study only on catch-up video sessions.

Implications with mobile access networks

In our dataset, clients had access to Internet via different technologies including EDGE, WCDMA (3G), HSPA (3G+), and, since January 2012, HSPA+ (3G++), which covers nearly 50% of the overall considered population of end users during the period of collection. In our dataset, we observe that 94.6% of the catch-up HAS sessions were streamed over 3G access networks, 4.8% from the EDGE radio access, while 0.06% of HAS sessions were streamed from both 2G and 3G access network, indicating that the corresponding clients were on the move when requesting chunks and experienced an inter-system handover during their HAS session. Actually the data transmission performance depends on multiple parameters, including the user's distance from the cell towers, the coding and modulation schemes used by the users and whether the User Equipment (UE) is compatible with the mobile access network. For example, a given UE may be compatible only with the earliest release of 3G, while being in an area covered by WCDMA and HSPA; in that case the UE selects the WCDMA network. All these constraints may influence the QoE and the chunk download time.

Then, we differentiate streaming HAS contents from EDGE networks and from 3G radio access networks. In Figure 8 and 9, we compare each Average requested Encoding Bitrate (AEB) to the corresponding Average Download Throughput (ADT) of each video session. We then normalize them to match the scale ranging from 0 to 100 on the x-axis.

$$ADT = \frac{\sum_{profile=0}^{7} DT_{profile}}{\sharp requested\ profiles\ per\ session}$$

such as:

$$DT_{profile} = \frac{\text{Downloaded bytes per profile including packet header}}{\text{Time needed to download the corresponding bytes}}$$

$$AEB = \frac{\sum_{profile=0}^{7} EB_{profile}}{\sharp requested\ profiles\ per\ session}$$

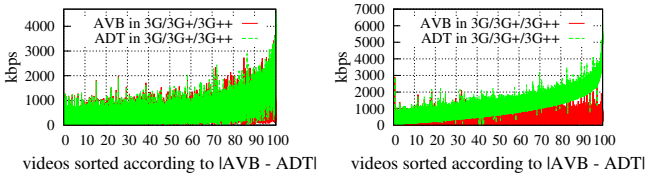
such as:

$$EB_{profile} = \frac{\text{Downloaded bytes per profile without packet header}}{\sharp \text{ chunks per profile * chunk duration}}$$

To emphasize the difference between AEB and the corresponding ADT of each session, we sort video sessions according to the difference $|ADT - AEB|$. In this figure we define a new metric:

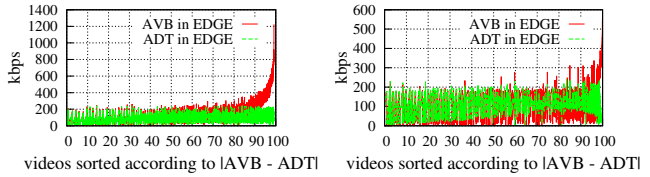
$$f_{buffer} = \frac{\text{session duration}}{\sharp \text{ chunks per session * chunk duration}}$$

if ($f_{buffer} > 1$), then the playback was potentially interrupted at least one time during the stream, as if the time spent in the network to deliver all chunks exceeds the playing time, this would certainly turn the client-player into the buffering mode. We show in table IV the average, median and CoV of all measured ADT and AEB for both 2G and 3G/3G+/3G++ networks with respect to f_{buffer}. Clearly 2G access network restricts requesting higher presentations as in average the Download throughput is less than 113 kbps. This raises the question of the efficiency of the rate adaptation for regions with limited penetration of high speed mobile networks and sophisticated smart phones such as developing countries.



(a) $f_{buffer} > 1$ (b) $f_{buffer} < 1$

Figure 8. AEB versus ADT in UMTS



(a) $f_{buffer} > 1$ (b) $f_{buffer} < 1$

Figure 9. AEB versus ADT in EDGE

	WCDMA/HSPA/HSPA+				EDGE			
	$f_{buffer} < 1$		$f_{buffer} > 1$		$f_{buffer} < 1$		$f_{buffer} > 1$	
	ADT(kbps)	AEB(kbps)	ADT(kbps)	AEB(kbps)	ADT(kbps)	AEB(kbps)	ADT(kbps)	AEB(kbps)
Average	1087.73	336.061	576.011	329.124	112.528	124.044	88.2743	142.803
Median	916.946	321.571	403.984	256.479	115.954	147.477	84.2004	119.748
CoV	1,5242	2,0078	1,06	1,2742	2,507	1,7403	2,0628	1,3518

Table IV
AVERAGE, MEDIAN AND COV OF ADT AND AEB IN 2G AND 3G NETWORKS

Content providers should give more attention, while setting the Manifest file and define more appropriate presentations to clients connecting from 2G access networks. In the collected data set we find that around 74.03% of sessions streamed over 2G networks experience $f_{buffer} > 1$. This means that these sessions certainly suffer from video interruptions. However clients with 3G connections exhibit a better QoE as the requested encoding bitrates are much higher than 2G.

Besides, in Figures 8 and 9 we observe that when $f_{buffer} > 1$, the requested encoding bitrates are close and even bypass the download throughput experienced by clients. Therefore this is a potential cause of buffer depletion and thus a bad quality of experience. However, when $f_{buffer} < 1$, the Download throughput is much higher than the requested encoding bitrates. We find that the average download throughput is around 1080 kbps. Therefore clients are able to request higher quality presentations.

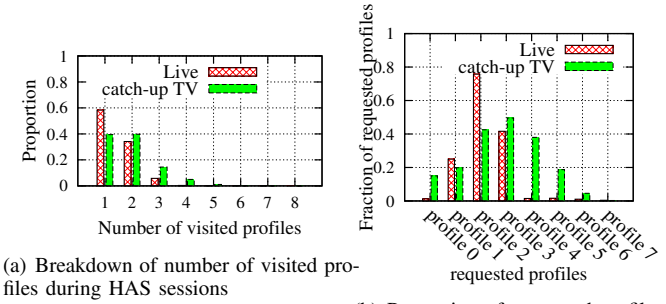
In Table IV, we observe that the CoV of AEB when $f_{buffer} < 1$, which represents 73.43% of HAS sessions when streaming over 3G access networks, is extremely high (>2). In that case, clients experience a wide range of download throughputs as different UMTS releases are being deployed by the telecom operator in France. Therefore clients are more likely to request chunks from different profiles. We provide deeper analysis on how client requests different qualities of HAS contents in Section V.

V. Profiles in HAS

A. Distribution of requested profiles in HAS sessions

Figure 10(a) shows the distribution of the number of visited profiles during the HAS sessions from the collected dataset. Almost 80% of catch-up TV HAS sessions visited only two different profiles, 40% never switched to a different profile. This is largely due to the significant proportion of very short catch-up TV sessions as shown in Figure 3(a): the HAS session is ended by the user before any HAS adaptation happens. This phenomenon is even more pronounced in the case of the live TV HAS sessions. There is no HAS adaptation for almost 60% of the live TV HAS sessions, thus 20% more than in the catch-up TV HAS sessions. This is most probably due to the aforementioned specific behavior of the users watching live TV: they access to a TV channel without necessarily knowing what is currently broadcasted, and they need a few seconds to decide whether to keep or stop watching it. Previous studies estimated that the delay between an event that triggers HAS adaptation and the effective transition to another representation level is about 14 seconds [11], which is largely superior to the time needed by the user to recognize the TV program and to decide to stop watching it. Therefore these clients abort their sessions before any transition gets triggered.

Figure 10(b) depicts the requests' distribution per profile respectively for the live TV HAS sessions and for the catch-up TV HAS sessions. First, we note that the sum of the fractions of the requests as per the 8 profiles is higher than one (both in the cases of the live TV HAS sessions and catch-up TV sessions), since, as shown on Figure 10(a), several profiles are visited during a significant part of the HAS sessions (40% of the live TV HAS sessions and 60% of the catch-up TV HAS sessions). We also observe that the clients' requests are more concentrated in profiles 2, 3 and 4 for catch-up HAS sessions. However, there are some significant differences between the live and catch-up TV HAS sessions. The proportion of live TV HAS sessions visiting profiles above profile 3 is marginal. This calls for deeper investigations on how to best manage the specificities of live TV HAS sessions in a mobile context, in the content preparation process (at the time the chunks are generated and the manifest files, updated dynamically) as well as in the delivery process either for Live or catch-up sessions, for example with specific caching strategies contents and for the most frequently visited profiles (*i.e.* profiles 2 and 3 for Live sessions and 2, 3 and 4 for catch-up sessions).



(a) Breakdown of number of visited profiles during HAS sessions

(b) Proportion of requested profiles

Figure 10. Number of visited profiles and profiles selection during HAS sessions

Finally, for catch-up HAS sessions, profiles 5, 6 and 7 corresponding to the encoding bitrates higher than 600 kbps, are rarely visited. The most probable reason for this behavior is that most of TV broadcasters set a range of encoding bitrates that is less than 600 kbps. However with an HSPA or HSPA+ connection, clients could be eligible to request such profiles. Besides, Figure 8(b) shows that in an important number of sessions, clients exhibit a download throughput higher than 600 kbps (profile 5). We believe that adding higher profiles within the manifest file with higher encoding bitrates would not necessarily cause performance degradation at the client side especially with the deployment of the LTE technology and with the on-going improvement in the computing capabilities at the client devices. At this time, we expect that this will become more challenging for the service providers as to deliver contents with such large ranges of encoding bitrates.

	Nchunks /session	Nchunks /profile 0 /session	Nchunks /profile 1 /session	Nchunks /profile 2 /session	Nchunks /profile 3 /session	Nchunks /profile 4 /session	Nchunks /profile 5 /session	Nchunks /profile 6 /session	Nchunks /profile 7 /session
Max	347.42	35.92	189.34	198.63	174.60	215.77	83.51	111.46	20.43
$Average$	58.40	4.25	23.88	29.29	12.77	45.19	20.88	13.77	6.92
$Median$	41.43	2.24	10.14	20.64	5.93	31.26	16.35	7.7	5.32
$CoV = \frac{\sigma(deviation)}{\mu(mean)}$	1.03	0.53	1.21	1.15	1.56	0.96	0.74	0.96	0.55

Table V
BREAKDOWN OF NUMBER OF CHUNKS BY SESSION

B. Sojourn time per profile

In this part, we only focus on the catch-up TV HAS sessions, as we capture a significant variation in requested profiles: Client requests are distributed within several bitrates, unlike Live sessions where chunks requests belong most probably to either profile 2 or 3.

In Table V, we show the breakdown per profile of the max, mean, average and Coefficient of Variance (CoV) of number of chunks per video content and relative to each profile. We only considered the 17% of most popular videos, as they account for more than half of the overall requests. More precisely, we focus on videos that were requested at least 500 times during the measurement period.

The mean and median requested number of chunks corresponding to profile 4 is the highest one with CoV lower than one. This means that clients request approximately the same number of chunks when visiting this profile and that the corresponding video bitrate of their profile matches well the bandwidth and CPU capabilities of the clients. However, Content providers may also restrict the encoding ranges so that clients would see profile 4 as the highest available profile in the manifest file.

We also observe that although profile 3 is highly requested by clients, as shown in Figure 10(b), we find that the average and median requested number of chunks by session in profile 3 is low comparing to profiles 2, 4 and 5. This means that usually clients do not sojourn for a long time in profile 3. They rather decide to switch to other profiles, mainly profile 2, 4 or 5. The reason behind this is that in HLS, the first requested profile is preset in the manifest file [12]. All clients requesting a given video start by requesting the first chunk in that preset profile. We show in Table VI that more than half of video contents within our dataset are preset with profile 3 as a Starting Profile (SP). Therefore, when requesting these videos, clients' players start buffering the first chunks (typically 3 chunks) from that preset profile and then decide to switch to other profiles.

	SP_0	SP_1	SP_2	SP_3	SP_4	SP_5	SP_6	SP_7
Stats	4.4%	0.6%	27.98%	52.72%	4%	9.7%	0.6%	0%

Table VI
BREAKDOWN OF STARTING PROFILE SP_i

This is confirmed in Table VII, which shows the breakdown of the Number of Visited Profiles (VP) during a HAS session assuming its Starting Profile is SP_i. In this table, more than 80% of the HAS sessions for which the starting profile is preset to 3 visit several other profiles.

	SP_0	SP_1	SP_2	SP_3	SP_4	SP_5	SP_6
1 VP	43.51%	21.58%	28.99%	19.93%	62.21%	55.4%	79.3%
2 VP	26.85%	53.02%	49.05%	59.21%	28.09%	26.59%	11.2%
3 VP	20.15%	18.6%	18.26%	14.57%	7.82%	12.81%	4.85%
4 VP	6.37%	6.5%	3.4%	6.09%	1.5%	4.6%	3.3%
5 VP	2.12%	0.3%	0.3%	0.2%	0.4%	0.6%	1.4%

Table VII
BREAKDOWN OF NUMBER OF VISITED PROFILES (VP) ASSUMING STARTING PROFILE SP_i

To conclude this section, the key outcomes of this analysis per profile are the followings:

- In HLS, the first requested profile is preset in the manifest file; in our trace about 80% of HAS sessions start with either Profile 2 or 3 (as shown on Table VI).
- While Profile 3 is among the most-frequently preset profiles in the manifest files, the sojourn time in Profile 3 is relatively short. The clients rather decide to switch quickly to other profiles, mainly profile 2, 4 or 5.
- In contrast, when clients start with the highest profiles (> 3), they rarely make transitions. They experience the best and most adequate bitrate level during their whole HAS session. For example, 79.3% of the catch-up sessions starting at Profile 6 show no transition.

VI. RELATED WORKS AND CONCLUSION

Most of the related works were already cited through the analysis and comparisons that we made in the previous sections. We briefly summarize here the other most relevant related works. In [13], authors analyzed a large-scale Chinese TV service. They analyzed channels popularity and compared the access frequencies of mobile TV channels against IPTV systems. In contrast, we have collected all HAS traffic of all TV service providers within France and we delved the content popularity at the chunk level. In [14], authors made comparison between the three existing types of streaming: Progressive download, Progressive download with byte range, and HLS. However they did not provide

a thorough analysis about the profile-based analysis as we reported in our present paper.

One of the main lessons of this analysis is that caching in the network all bitrates of all chunks of all videos is clearly not the best solution. This is particularly true for the considered content is subject to flash crowds, where the cache will be under a heavy workload. We started to show the considerable gain in update-ratio and per chunk position hit ratio when setting the appropriate thresholds. In future works, we plan to concentrate on the adaptation process by tracking the exact timestamp of clients when switching between profiles in each session. We will also study the implication of such transitions on the cache eviction behavior and propose novel caching policies tailored for the HTTP bitrate adaptation.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2011 – 2016," Tech. Rep., 2012.

[2] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 1–14.

[3] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an ip network," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008, pp. 71–84.

[4] "http://developer.apple.com/library/ios/technotes/tn2224." [Online]. Available: http://developer.apple.com/library/ios/technotes/tn2224/index.html

[5] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 314–329.

[6] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable?" *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 133–144, 2007.

[7] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min, "Inside the bird's nest: measurements of large-scale live vod from the 2008 olympics," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 442–455.

[8] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, Oct. 2006, p. 333.

[9] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," *ACM MMSys*, vol. 11, pp. 157–168, 2011.

[10] B. F. Hong D, De Vleeschauwer D, "A chunk-based caching algorithm for streaming video," in *Proceedings of the 4th Workshop on Network Control and Optimization*, 2010, pp. 33–40.

[11] L. D. Cicco and S. Mascolo, *An Experimental Investigation of the Akamai Adaptive Video Streaming*, 2010.

[12] http://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/streamingmediaguide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html/appleref/doc/uid/TP40008332-CH102-SW1.

[13] Y. Li, Y. Zhang, and R. Yuan, "Measurement and analysis of a large scale commercial mobile internet tv system," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 209–224.

[14] J. Erman, A. Gerber, K. Ramakrishnan, S. Sen, and O. Spatscheck, "Over the top video: The gorilla in cellular networks," *ACM New York*, 2011.